

Trilaminar Multiway Reconstruction Tree for Efficient Large Scale Structure from Motion

Kun Sun, Wenbing Tao

National Key Laboratory of Science and Technology on Multi-spectral Information Processing
School of Automation, Huazhong University of Science and Technology
Wuhan, 430074, China

sunkun@hust.edu.cn, wenbingtao@hust.edu.cn

Abstract

Accuracy and efficiency are two key problems in large scale incremental Structure from Motion (SfM). In this paper, we propose a unified framework to divide the image set into clusters suitable for reconstruction as well as find multiple reliable and stable starting points. Image partitioning performs in two steps. First, some small image groups are selected at places with high image density, and then all the images are clustered according to their optimal reconstruction paths to these image groups. This promises that the scene is always reconstructed from dense places to sparse areas, which can reduce error accumulation when images have weak overlap. To enable faster speed, images outside the selected group in each cluster are further divided to achieve a greater degree of parallelism. Experiments show that our method achieves significant speedup, higher accuracy and better completeness.

1. Introduction

Reconstructing 3D models using online images is a challenging task due to the large scale image set, unknown scene overlap and uncalibrated camera parameters. Such kind of images are usually reconstructed with the Structure from Motion (SfM) method. The most common SfM method operates in an incremental fashion, which consists of three steps: 1) Image matching. In this step, features are extracted and matched between images. Afterwards geometry verification is performed to remove bad matches. 2) Initial model reconstruction. Two starting images having the largest number of matches, subject to the condition that they can not be well modeled by a single homography are selected to build the initial model. 3) Incrementally adding new images. The pose of a new camera is estimated by solving the Perspective-n-Point (PnP) [6] problem and then refined with the bundle adjustment algorithm [17].

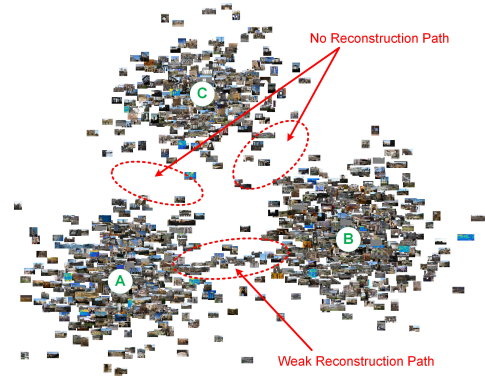


Figure 1. A common situation in large scale SfM with Internet images. Images are dense at A, B and C but sparse at other places. A and B are connected by a weak reconstruction path. But there is no sufficient overlap between AC and BC.

While the above pipeline has broad applications on small and medium problems, it is awkward when dealing with large image sets. Fig. 1 shows a common situation in large scale SfM with Internet photos. Images are dense at places A, B and C but sparse at other places. A and B are connected by a reconstruction path, which is composed of a series of overlapping images between them. But such a reconstruction path is missing between AC and BC. The performance of traditional methods is largely affected by the uncertainty of starting point selection. For example, if the starting point is selected in A, both A and B could be reconstructed. However, there might be large accumulation error since the overlap between them is weak. If the starting point is found in C, neither A nor B could be reconstructed. Some existing systems [21, 11] tackle this problem by restarting a new SfM procedure from the remaining images. However, good models might be reconstructed after many failures, which wastes a lot of time. To overcome these shortcomings, some methods divide the original image set and reconstruct each part

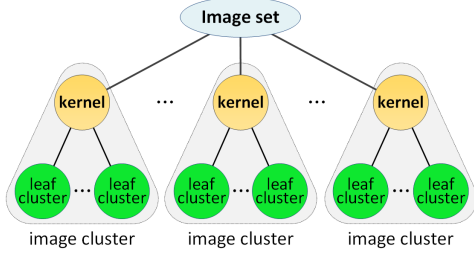


Figure 2. The Trilaminar Multiway Reconstruction tree (TMR-tree).

independently. The images can be clustered according to their location manually. But this is difficult for Internet images because they are totally unordered and not all of them are tagged with geo-information. Or one can find connected components in the image matching graph [5]. In this case, A and B are in the same component and the accumulation error will not be eliminated. Normalized Cuts is used to partition the image set automatically in [7]. But the number of clusters must be specified by the user, which is hard to know in advance. Some other methods extract iconic images [7, 3, 5, 11] or skeletal graphs [15, 1] from the original image set to speed up the reconstruction. However, error accumulation and reconstruction interruption may still happen if the starting point is not well selected after image sampling.

In this paper, we propose a new method for automatic data partitioning and multiple proper starting points selecting. The partitioning result is described with a Trilaminar Multiway Reconstruction tree (TMR-tree), which is shown in Fig. 2. A partition suitable for reconstruction must satisfy two requirements. On the one hand, each partition should contain a set of images having large overlap between each other. The reconstruction will start from these images to ensure accuracy. On the other hand, any two images in the same partition should be connected by scene overlap, so that all of them could be added. To this end, our method partitions the image set in two steps. We first search for places where images are densely distributed. Several kernels are found at these places, shown as yellow nodes in Fig. 2. Each kernel contains a few images having large overlap with each other. The number and size of kernels should not be too large. Images outside the kernels are called leaves. Next, all the images are clustered according to their optimal reconstruction paths to the kernels. Each cluster consists of two parts: one kernel and a set of leaves around it. Accordingly, the reconstruction performs in a hierarchical way. In the first stage, all the kernels are reconstructed in parallel to build base models of the scene. In the second stage, the leaves are added to these base models. Since kernels only take a small part of the whole image set, the number

of leaves in a cluster might still be large. To enable faster speed, these leaves are further split into leaf clusters, which are green nodes in Fig. 2. Each leaf cluster can be independently added to the same base model in parallel without distinct accuracy deterioration. The models of leaf clusters sharing the same base model are merged to get the model of an image cluster. Then models of different image clusters are merged to a complete one.

2. Related Works

Large scale structure from motion has witnessed great development in recent years. Several complete structure from motion systems have been proposed. Snavely *et al.* [16, 14] are among the first to propose a complete incremental SfM pipeline. The backbone of their *Photo Tourism* system is a structure from motion approach which computes the photographers' locations and orientations, along with a sparse 3D point cloud.

Li *et al.* [8] proposed to capture the major aspects of the scene using an iconic scene graph. Their method divided images into small clusters. Image matching and geometry verification are only performed between images within the same cluster. Each cluster is represented by an iconic image. In order to make SfM perform more efficiently, they partitioned the iconic scene graph with normalized cuts [13] and run incremental SfM on each part.

Agarwal *et al.* [1] designed a system running on a collection of parallel distributed machines to efficiently reconstruct a city. They paid a lot of effort to reduce the cost of scheduling between different tasks. They computed a skeletal set of photographs [15] instead of reconstructing all the images. Frahm *et al.* [3] improved the work of [1] by reconstructing a city on a single machine with multi-core CPUs and GPUs. They concatenated the global GIST descriptor [10] with a subsampled image. Then the descriptor was compressed to shorter binary code so that it is memory efficiency for GPU computation. They also generated dense 3D model using fast plane sweeping stereo and efficient depth map fusion algorithms.

Wu [22] proposed a new SfM framework that has $O(n)$ time complexity. He used top-scale feature matching to coarsely identify image overlapping, which saved much time in image matching. During reconstruction, his method performed full bundle adjustment optimization after the model increases a certain ratio and partial bundle adjustment on a constant number of recently added cameras to reduce the accumulated time of bundle adjustment.

Shah *et al.* [12] proposed a coarse-to-fine SfM strategy. In the first stage a coarse yet global model is quickly reconstructed using high scale SIFT features. This model offers useful geometric constraints for the second stage, in which the model is enriched by localizing remaining images and triangulating remaining features.

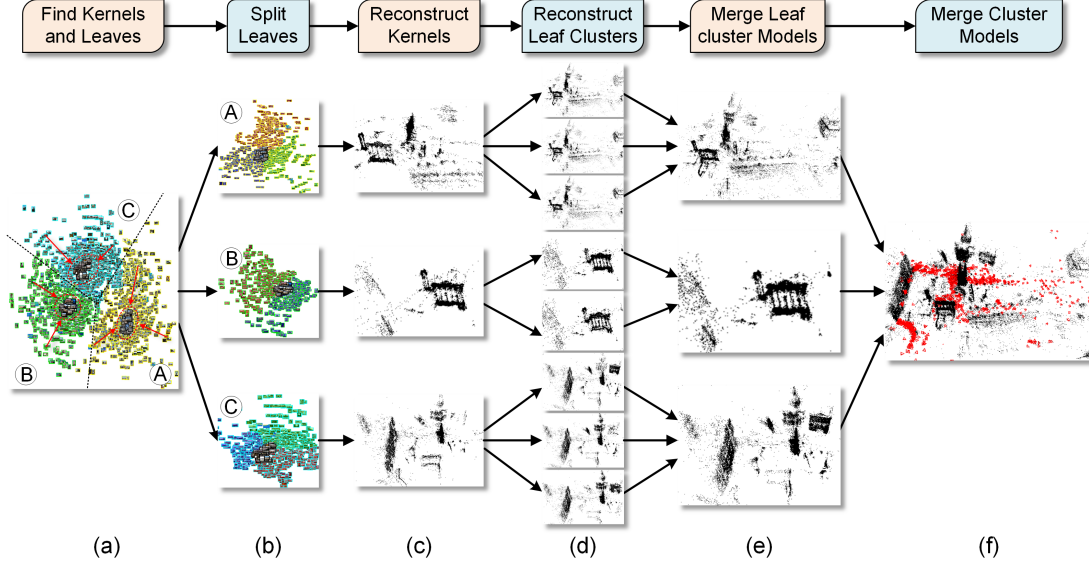


Figure 3. The overall flowchart of our method. (a) The whole image set is divided into three clusters A, B and C. In each cluster the kernel is drawn with black boxes. The leaves in each image cluster are drawn with yellow, green and cyan boxes, respectively. (b) Leaves in an image cluster are further split into leaf clusters. (c) The base models reconstructed from the kernels. (d) Reconstruction results after adding each leaf cluster to the same base model. (e) The model of each image cluster after merging the models in (d). (f) The final result after merging the models of different image clusters.

Heinly *et al.* [5] advanced the state-of-the-art SfM methods from city-scale modeling to world-scale modeling on a single computer. They also leverage the idea of iconic images to represent small image clusters. The database-side feature augmentation is applied so that an iconic image can cover a broader set of views. For the ability to handle world scale images, their system stores an image’s data in memory only when it is needed.

The latest achievement in large scale SfM is reported in [11]. Their work improved several components of the state-of-the-art methods, such as geometry verification, view selection, triangulation and bundle adjustment to make a further step towards a robust, accurate, complete and scalable system.

3. Overview

In this section, an overview of the proposed method is given. Before our algorithm starts, some preparations such as feature extraction, image matching and matching graph construction are made. Suppose we have a set of unordered images $I = \{I_i\}_{i=1}^N$. The SIFT [9] features are extracted from each image. Each image is matched to its top K nearest neighbors searched from a trained vocabulary tree. The value of K is set to 30 according to other reported papers. A faster GPU implementation [20] is adopted to speedup the burdensome matching procedure. Wrong matches are removed by estimating the epipolar geometry between two views using the RANSAC [2] algorithm.

After image matching is done, the matching graph is constructed. The matching graph $G < V, E >$ is an undirected weighted graph with a set of vertexes V and edges E . A vertex $v_i \in V$ represents an image. If two images have scene overlap, an edge is added between the corresponding vertexes. We build two kinds of matching graphs: a similarity graph S and a difference graph D . They have the same number of vertexes and edges, but the meaning of their edge weights are different. In the similarity graph S , the edge weight s_{ij} reflects the content similarity between two images. An intuitive way is to measure this similarity with the number of matches between two images. However, this measurement is sensitive to image resolution and texture. High resolution or textured images will have more matches than low resolution or less textured images. In this paper, s_{ij} is computed from the following formulation:

$$s_{ij} = \frac{n_{ij}}{n_i \cup n_j}, \quad (1)$$

in which n_{ij} is the number of matches between two images I_i and I_j , n_i and n_j are the number of feature points on image I_i and I_j that have corresponding points on the other images, respectively. Eq. (1) is also known as the Jaccard similarity coefficient. A larger s_{ij} indicates that I_i and I_j have more scene overlap. It is robust to different image sizes and scene textures. The weight of the difference graph D is then computed from:

$$d_{ij} = 1 - s_{ij}. \quad (2)$$

The flowchart of the proposed method is shown in Fig. 3. In Fig. 3(a) three kernels are found and all the images are divided into three clusters A, B and C according to their optimal reconstruction paths to the kernels. Each cluster contains a kernel and some leaves around it. The kernels are drawn with black boxes and the leaves in different clusters are in yellow, green and cyan, respectively. Fig. 3(b) shows that the leaves in an image cluster are split into several leaf clusters to acquire faster reconstruction speed. The reconstruction path from each image in a leaf cluster to the kernel should lie within the same leaf cluster. Thus, each leaf cluster could be independently added to the same base model in parallel without distinct accuracy deterioration. After finishing the above steps, the kernels are first reconstructed in parallel to get several base models of the scene, which are shown in Fig. 3(c). These base models are reliable because they are reconstructed from images with large overlap. Next, different leaf clusters in one image cluster are independently added to the same base model in parallel, and the results are shown in Fig. 3(d). The base models are enriched by adding images in the leaf clusters in this step. Since these models share the same base model, it's easy to merge them together to get the model of each image cluster. The models for cluster A, B and C are shown in Fig. 3(e). Finally, the models of different image clusters are merged to a complete one, which is shown in Fig. 3(f).

4. Trilaminar Multiway Reconstruction Tree

The data partitioning result is modeled by a Trilaminar Multiway Reconstruction Tree (TMR-tree). The top layer is a single root node representing the whole image set. The nodes in the middle and bottom layers correspond to kernels and leaf clusters. In this section, the method for building the Trilaminar Multiway Reconstruction Tree is introduced. The steps include: finding kernels, image clustering and finding leaf clusters.

4.1. Finding Kernels with A Multi-layer Greedy Strategy

Kernels are used to reconstruct base models of the scene. They should be found at places where images are densely distributed. Images at such places have large scene overlap between each other, so that the base model reconstructed is accurate in precision, representative and centric in location. Since there is no absolute standard for judging whether the distribution of the cameras is dense, we choose a loose greedy manner to progressively find multiple kernels from the whole image set.

Since the mission of a kernel is to reconstruct an initial local model of the scene, it is not expected to contain too many images. Suppose the ideal size of a kernel is between m and $\alpha \cdot m$, where m is a positive number and $\alpha \geq 1$ is an inflation factor. We adopt a greedy strategy to find ker-

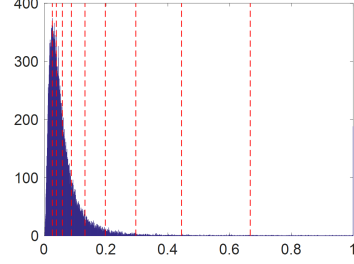


Figure 4. The distribution of edge weights in the similarity graph and θ_i .

nels in a layered graph. Given the number of layers k and a set of edge weight thresholds $\theta_i (i \in 1, 2, \dots, k)$ satisfying $\theta_i > \theta_{i+1}$, the edges whose weights are greater than θ_i are added back to the similarity graph S in the i^{th} step. Then we find connected components in the graph. If none of them is larger than m , we continue by reducing the edge weight threshold and adding more edges in the next step. If a connected component is larger than m but smaller than $\alpha \cdot m$, the images in this component form a new kernel and the corresponding vertexes are removed from the current graph. If a connected component is larger than $\alpha \cdot m$, we will find kernels with proper size in this component recursively using the same method above. In this way, we can guarantee that each kernel is a set of images that have the strongest overlap among the current remaining images.

Computing θ_i is an important problem. Denote the minimum and maximum edge weights in the similarity graph as a and b , respectively. In practice, a is set to a value ε larger than the minimum of the edge weights so that images having too weak overlap with others are not considered in this stage. The range $[a, b]$ is divided into k intervals and edges are added from higher interval to lower interval. Fig. 4 shows the distribution of all the edge weights in the similarity graph. It can be seen that there is an obvious peak near 0.02. If the intervals are divided uniformly, the higher intervals contain few edges but the lower intervals contain numerous edges. As a result, it is difficult to find large enough kernels at the first few steps but will soon fall into deep recursion because adding a great many edges will make the connected component grow fast. In this paper, θ_i is computed from the following formulation:

$$\theta_i = a + \frac{b-a}{1.5^{i-1}}, i \in 1, 2, \dots, k, \quad (3)$$

which are red vertical dash lines in Fig. 4. It can be seen that such a division can keep the number of edges in each interval roughly the same.

Once several kernels have been found, an exemplar image which will be used as the starting point is found in each kernel. It should have dense overlap with other images so that the initial model can easily spread the 3D structure to

nearby space. The Affinity Propagation (AP) clustering algorithm [4] is applied to images in each kernel. All the centers and their adjacent neighbors on the similarity graph \mathbf{S} are treated as the candidates for the exemplar image. Affinities between data points required by AP clustering are computed from Eq. (1). The reason for choosing AP clustering has two aspects. On the one hand, AP clustering algorithm can automatically determine the number of clusters. On the other hand, the center of a cluster is one data point instead of a virtual mean position. For each candidate image, the following score is computed:

$$\delta(v) = h_{deg}(v) + \beta_1 \cdot h_{sim}(v) + \beta_2 \cdot h_{ndeg}(v). \quad (4)$$

The first term $h_{deg}(v)$ is the degree of the vertex v , which counts the number of images that overlap with it. The second term $h_{sim}(v)$ is the average similarity from the vertex v to its neighbors, namely the mean adjacent edge weight on \mathbf{S} . This term encourages the vertex v to have large overlap with its neighbors. The last term $h_{ndeg}(v)$ is the average degree for the neighbors of v . That is to say, not only v itself should overlap with many images, but also the images overlapping with it should also overlap with as many other images as possible. This strengthens the potential of the starting point to build an accurate initial model and spread 3D structure to nearby space. Image with the highest score is selected as the exemplar image.

4.2. Clustering Images According to Their Optimal Reconstruction Paths to the Kernels

In this part, all the images are clustered by treating the kernels as centers. This is not a simple image classification problem because the clusters may not be good for reconstruction. Hence, in our method all the images are clustered according to their optimal reconstruction paths to the kernels. A reconstruction path is composed of a series of overlapping images which can pass the 3D structure. There can be multiple reconstruction paths from an image to a kernel. We think the optimal reconstruction path should consist of a series of largely and equally overlapping images. In other words, the maximum difference between adjacent images on an optimal reconstruction path should be minimized, which is shown in Fig. 5. In this example, two reconstruction paths between images A and B are shown. The edge weights reflects the difference between two images. The red path has shorter length than the green path. However, it is not considered as the optimal reconstruction path because its edge weights vary a lot. There is an edge whose weight (0.66) is much larger than the other two edges (0.14 and 0.18). This means that the 3D structure has to be propagated via relatively weak image overlap, which is unreliable. Although the length of the green path is a bit longer than the red path, its edge weights are similar. If the green path is selected as the optimal reconstruction path, the risk



Figure 5. Find the optimal path (in green) using the Multi-layer Shortest Path (MSP) algorithm.

of passing 3D structure via weak overlap will no longer exist.

In this paper, a Multi-layer Shortest Path (MSP) algorithm is proposed to find the optimal reconstruction paths from each image to the kernels. Our MSP algorithm operates on the difference graph \mathbf{D} , in which the edge weight indicates the scene difference between images. This graph is divided into L layers by a set of increasing weight thresholds $\phi_t (t \in 1, \dots, L)$ satisfying $\phi_t < \phi_{t+1}$. More specifically, the range of edge weights $[\min(d_{ij}), \max(d_{ij})]$ on \mathbf{D} is divided into L homogeneous intervals. For each interval the step length is $l = (\max(d_{ij}) - \min(d_{ij}))/L$ and ϕ_t is computed from

$$\phi_t = t * l + \min(d_{ij}), t = 1, \dots, L. \quad (5)$$

Edges whose weights are smaller than ϕ_t are added back to \mathbf{D} in the t^{th} layer. Denote w as a leaf. The shortest paths from w to the exemplars of the kernels are computed. At the very beginning, no paths exist between w and the kernels. If none of the paths to the kernels are found in the t^{th} layer, we then add more edges by using a larger edge weight threshold in the next layer. With more and more edges are added, the paths between w and the kernels will be found. The optimal path between w and a kernel is the path found for the first time. Although in the following layers the paths between w and the same kernel will also be found, they are not optimal. In the example of Fig. 5, the optimal path in green will be found before the path in red. If the optimal paths to different kernels are found in the t^{th} layer, then w is assigned to the cluster of the kernel with the smallest path length. Once a leaf has been clustered, it will not be processed in the following layers.

4.3. Finding Leaf Clusters using Radial Agglomerate Clustering

Since kernels take only a small part of the image set, the number of leaves in an image cluster might be still too large. Adding them sequentially to the base model will be time consuming. Thus, they are further divided into leaf clusters to achieve faster speed. Three conditions should be satisfied so that each leaf cluster could be reconstructed in parallel without distinct accuracy deterioration. (1) Images within each leaf cluster should have considerable overlap with each

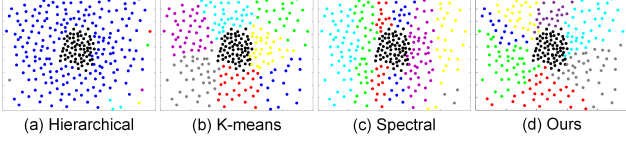


Figure 6. The results of Hierarchical clustering, K-means clustering, Spectral clustering and the proposed Radial Agglomerate Clustering (RAC) algorithm on 2D synthetic data. The black points in the center are manually selected kernel points.

other. (2) Each leaf cluster should have strong overlap with the kernel so that it can be added to the base model. (3) The size for these leaf clusters should be balanced to reduce waiting time of different threads.

In this paper, an improved Radial Agglomerate Clustering (RAC) algorithm is proposed to divide leaf clusters. The distance between leaves in an image cluster is the shortest path length on the subgraph formed by this image cluster. The distance between two leaf clusters is measured by the distance of two closest images between them. The number of leaf cluster K_c is computed from $K_c = \text{round}(\frac{M}{e})$, where M is the number of leaves in an image cluster and e is the ideal mean size of each leaf cluster. We expect a leaf cluster to be larger than the kernel and set $e = r \cdot m$, where r is a positive integer. At the very beginning, each leaf in the image cluster is an isolate leaf cluster. In each step a score is computed for ever possible leaf cluster pair and two leaf clusters with the smallest score are merged until K_c leaf clusters remain. The score is computed from:

$$\varphi(p) = \sigma_1 \cdot g_d(p) + \sigma_2 \cdot g_k(p) - \sigma_3 \cdot g_r(p) + \sigma_4 \cdot g_c(p) \quad (6)$$

where p is a possible leaf cluster pair composed of two leaf clusters c_1 and c_2 . $\sigma_1, \sigma_2, \sigma_3$ and σ_4 are four positive tuning parameters. The first term $g_d(p)$ is the distance between c_1 and c_2 , which prefers to merge close leaf clusters to meet requirement (1). The second term $g_k(p)$ measures the distance between the kernel and the leaf cluster after merging c_1 and c_2 . This term encourages that after merging two leaf clusters the new one has strong connection with the kernel. The third term $g_r(p)$ is the difference between the distances from the two leaf clusters to the kernel. It will guide the merging along the radial direction. The second and third terms act together to meet requirement (2). The last term $g_c(p)$ counts the cardinality of c_1 and c_2 after merging them to a new one. It will tend to merge small leaf clusters at each step so that our final leaf clusters are balanced in size, which satisfies requirement (3).

An example showing the clustering results of our RAC algorithm and several other methods such as Hierarchical clustering, K-means clustering and Spectral clustering on synthesized 2D points is in Fig. 6. Kernel points are in the center with black color. The remaining points are divided

into 7 clusters by different methods. It can be seen that the result of hierarchical clustering is unbalanced. Some small clusters are far from the kernel points. The K-means clustering algorithm produces nearly balanced clusters. But the green and blue clusters are not adjacent with the kernel points. Similar problem happens to the spectral clustering algorithm as well. Our RAC method can produce radial, compact and balanced clusters.

5. Parallel Reconstruction with the TMR-tree

The reconstruction performs in two stages. In the first stage, the kernels are reconstructed in parallel to get several base models. A kernel is reconstructed from two initial images. The first image is fixed to the exemplar image found in Sec. 4.1. The second image is set to the one having the most matches and relatively wide baseline with the first image. In the second stage, leaf clusters of a kernel are added to the same base model, producing several individual models in parallel. The pose of each image is initialized by solving the PnP problem and then refined via bundle adjustment. At last, the individual models are merged in two steps. First, the models of different leaf clusters sharing the same kernel are merged to get the model of an image cluster. Next, different image cluster models returned in the first step are merged to get a complete model of the scene.

The reconstruction can be very fast if we have enough CPU cores and GPU cards because all the kernels and leaf clusters can be reconstructed in parallel. The complexity of our method is relevant to the kernel size $\alpha \cdot m$ and the leaf cluster size $r \cdot m$, which is $O(m)$. While even a linear-time SfM algorithm [22] has a complexity of $O(N)$, where N is the number of cameras. Our method has a theoretical speedup factor of $\frac{N}{m}$. If m increases, the complexity will increase but the models are more stable. On very large image set, the difference between N and m is large and the speedup is more obvious.

A similarity transformation is computed to merge two models. One of the difficulties is to detect the common parts between them. Since the same track reconstructed in different models may be inconsistent, directly finding common 3D points between models according to shared tracks will include a very large portion of outliers. In this paper we narrow down the number of suspicious common 3D points by the following method. Consider two models M_1 and M_2 , our method first finds an image in M_2 who has the most tracks reconstructed in M_1 . Then the tracks on this image who have also been reconstructed in M_2 are counted. If the number of such tracks is greater than a threshold τ , a Least-Square method [18] is implemented in the RANSAC framework to robustly estimate the similarity transformation between them. Otherwise do not merge M_1 and M_2 .

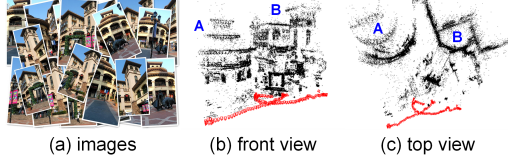


Figure 7. (a) The images captured in a street scene with a hand held digital camera. (b) Front view of its sparse 3D model. (c) Top view of its sparse 3D model.

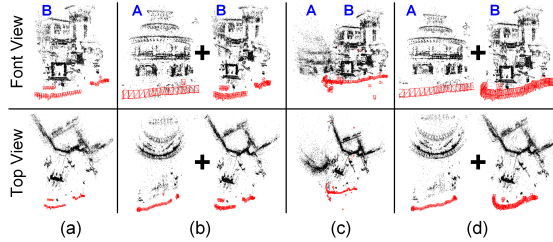


Figure 8. The reconstruction results of Bundler [14] and our method on self-captured images. (a) and (b) are results of Bundler and our method after removing 35 images. (c) and (d) are results of Bundler and our method after adding 21 images back.

6. Experiment Results

6.1. Parameter Settings and Implementation details

Among all the parameters, the most important two are m and ε because they affect the granularity of our data partition. A larger m will reduce the speed of the algorithm but the result might be more stable. A smaller m will result in fragmented partitions and inaccurate models. Similarly, increasing ε will reduce the number of kernels and reducing ε will result in more kernels. In this paper, we set $m = \min(70, 0.15 * Z)$, where Z is the total number of images, and $\varepsilon = 0.1$ empirically. For the rest parameters, we set $\alpha = 1.5$, $k = 15$, $\beta_1 = 100$, $\beta_2 = 1$, $r = 3$, $\sigma_1 = 1$, $\sigma_2 = 1$, $\sigma_3 = 3$, $\sigma_4 = 1$ and $\tau = 12$.

The GPU based bundle adjustment algorithm [23] is used. Our algorithm is implemented using C++ on Ubuntu 14.10 operating system. The experiments are tested on a machine with two Intel Xeon CPU E5-2630 v3 2.40GHz, one NVIDIA GeForce GTX TitanX graphics card and 256GB RAM.

6.2. Results on Self-captured Images

In this part we show the results on self-captured images. This image set contains 135 images of a scene beside the street. It contains two buildings A and B. When taking these images, we require adjacent pictures to have sufficient overlap so that all images could be used to reconstruct a complete model. Fig. 7 (a) shows some example images. The sparse 3D point cloud together with the poses of all the cam-

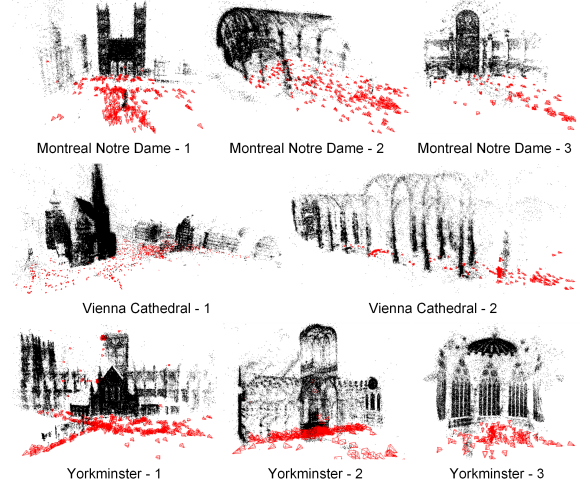


Figure 9. The reconstruction results of our method. From top to bottom are: 3 models in Montreal Notre Dame, 2 models in Vienna Cathedral and 3 models in Yorkminster, respectively.

eras are shown in Fig. 7 (b) and (c).

We first remove 35 images between A and B to cut the whole image set into two isolated parts. The reconstruction results for Bundler and our method are shown in Fig. 8(a) and (b), respectively. Our method can reconstruct two independent models while Bundler reconstructs only one of them. Then, 21 images are added back to the image set so that there is weak overlap between A and B. The result of Bundler in Fig. 8(c) shows that 3D structure is passed from B to A, and all the cameras are reconstructed. However, the structure and camera poses are wrong because the overlap between A and B are unreliable. In this case, it's better to build several good partial models from image subsets rather than build a wrong model with all the images. As is shown in Fig. 8(d), our method divides the image set into two clusters and builds two correct independent models for A and B. If more images in the middle are provided, the two independent models will be merged to a complete one in the right way.

6.3. Results on Public Benchmarks

Then the results on three public benchmarks including Montreal Notre Dame [19], Vienna Cathedral [19] and Yorkminster [19] are reported. The number of images in these datasets are 2298, 6288 and 3368, respectively. Images in these image sets are not connected and contains several independent primary models. Fig. 9 shows our reconstruction results on these datasets. We have found three primary models for Montreal Notre Dame, two primary models for Vienna Cathedral and three primary models for Yorkminster. Table 1 presents the partition results of our method on these image sets. The partition are done quickly within

Table 1. Partition result on the Montreal Notre Dame, Vienna Cathedral and Yorkminster image sets. For each dataset, the number of kernels, the number of leaf clusters belonging to a kernel and the time are given.

Dataset	Montreal Notre Dame				Vienna Cathedral					Yorkminster			
Kernels	K 1	K 2	K 3	K 4	K 1	K 2	K 3	K 4	K 5	K 1	K 2	K 3	K 4
Num Leaf Clusters	3	2	1	1	3	1	1	1	2	1	1	1	1
Time	7.127s				33.107s					48.324s			

Table 2. Results on the Montreal Notre Dame, Vienna Cathedral and Yorkminster datasets. For each model, the number of reconstructed cameras and the mean reprojection error are given. The running time for reconstruction is in the last column.

Dataset	Method	#Cameras			Error (pixel)			Time
Montreal Notre Dame	Ours	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	217.2s
		385	355	97	0.6241	0.7286	0.5112	
	VisualSfM	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	457s
		343	504	97	1.596	1.467	0.909	
	Bundler	-	399	-	-	1.5083	-	648.2s
Vienna Cathedral	Ours	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	337.4s
		1000	292		0.6550	0.8684		
	VisualSfM	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	1216s
		929	275		1.901	1.519		
	Bundler	1197	-		0.7106	-		12181.2s
Yorkminster	Ours	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	282.7s
		593	333	121	0.6935	0.5451	0.5905	
	VisualSfM	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	796s
		517	128	106	1.429	0.639	0.664	
	Bundler	-	-	122	-	-	0.6265	209.3s

1 minute. The number of kernels and leaf clusters for the three datasets are 4, 5, 4 and 10, 8, 4, respectively.

Our method is compared with two state-of-the-art methods: Bundler [14] and VisualSfM [21]. The number of reconstructed cameras, the mean reprojection error and the running time for reconstruction are given in Table 2. Since Bundler finds a single starting point and runs incremental SfM once, it can only reconstruction one of the models. The result of VisualSfM contains dozens of models because it iteratively runs a new incremental SfM in the remaining images after one model is reconstructed. However, most models are too small and only the primary models same with ours are considered here. The number of cameras reconstructed can reflect the model completeness of a SfM algorithm. For the total number of reconstructed cameras in each dataset, our method is the best. On a single model, the number of cameras reconstructed by us is similar with Bundler and larger than VisualSfM in most cases. The mean reprojection error indicates whether an algorithm is accurate. Our method achieves the smallest reprojection error on all the models.

The running time for reconstruction is in the last column of Table 2. It contains two parts: time for solving the PnP problem and time for bundle adjustment. For the first two datasets, Bundler takes the longest time because bundle adjustment is not performed on GPU. It runs the fastest on the third dataset because only a very small model is reconstructed. Both VisualSfM and our method can reduce much

time by using the GPU based bundle adjustment, while the PnP solver is still implemented on CPU. However, VisualSfM runs the PnP solver serially while our algorithm runs it on different kernels or leaf clusters in parallel. So our algorithm is 2-4 times faster than VisualSfM. Theoretically the GPU bundle adjustment can be parallelized on different kernels or leaf clusters as well. But our machine has only one GPU card. Hence, bundle adjustment is actually executed serially in our method. When dealing with very large image set containing hundreds of kernels and leaf clusters, if we have enough CPU cores and multiple GPU cards, all the kernels and leaf clusters could be truly reconstructed in parallel and the speedup will be more remarkable.

7. Conclusion

In this paper, an image set partitioning and starting point selecting method is proposed for efficient large scale SfM. The whole image set is divided into several clusters. Each image cluster consists of a kernel and a set of leaf clusters. A Trilaminar Multiway Reconstruction Tree (TMR-tree) is proposed to represent the partition result. The kernels are reconstructed first in parallel to build base models of the scene, and different leaf clusters of a kernel are added to the same base model simultaneously for parallel reconstruction. Experiments show that our method achieves much faster speed, more accurate poses and more complete models than state-of-the-art methods.

References

- [1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 72–79, Sept 2009.
- [2] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.
- [3] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 368–381. Springer Berlin Heidelberg, 2010.
- [4] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
- [5] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm. Reconstructing the world* in six days. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3287–3295, June 2015.
- [6] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epanp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.
- [7] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Proceedings of the 10th European Conference on Computer Vision: Part I, ECCV ’08*, pages 427–440, Berlin, Heidelberg, 2008. Springer-Verlag.
- [8] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5302 of *Lecture Notes in Computer Science*, pages 427–440. Springer Berlin Heidelberg, 2008.
- [9] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [10] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.
- [11] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [12] R. Shah, A. Deshpande, and P. J. Narayanan. Multistage sfm: A coarse-to-fine approach for 3d reconstruction, 2015.
- [13] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, Aug 2000.
- [14] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.
- [15] N. Snavely, S. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.
- [16] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *Proceedings of ACM SIGGRAPH, 2006*, pages 835–846, 2006.
- [17] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer Berlin Heidelberg, 2000.
- [18] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(4):376–380, Apr 1991.
- [19] K. Wilson and N. Snavely. *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, chapter Robust Global Translations with 1DSfM, pages 61–75. Springer International Publishing, Cham, 2014.
- [20] C. Wu. Siftgpu: A gpu implementation of scale invariant feature transform (sift). <http://cs.unc.edu/~ccwu/siftgpu/>, 2007.
- [21] C. Wu. Visualsfm: A visual structure from motion system. <http://ccwu.me/vsfm/>, 2011.
- [22] C. Wu. Towards linear-time incremental structure from motion. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 127–134, June 2013.
- [23] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064, June 2011.